

# More Examples

**FEST Assertions Module has moved to Github !**

**Check this page :<https://github.com/alexruiz/fest-assert-2.x/wiki>**

**The documentation below is for Fest 1.x which is no more maintained, we are focusing our effort to the 2.x version !**

...

The full example file is FestAssertExamples.java, you should be able to copy it in your IDE and run it successfully as JUnit tests.

```
import static org.fest.assertions.Assertions.assertThat;
import static org.fest.assertions.MapAssert.entry;

@Before
public void setup() {
    // let's do some team building :)
    fellowshipOfTheRing.add(frodo);
    fellowshipOfTheRing.add(sam);
    fellowshipOfTheRing.add(merry);
    fellowshipOfTheRing.add(pippin);
    fellowshipOfTheRing.add(gandalf);
    fellowshipOfTheRing.add(legolas);
    fellowshipOfTheRing.add(gimli);
    fellowshipOfTheRing.add(aragorn);
    fellowshipOfTheRing.add(boromir);
    // ring bearers
    ringBearers.put(Ring.nenya, galadriel);
    ringBearers.put(Ring.narya, gandalf);
    ringBearers.put(Ring.vilya, elrond);
    ringBearers.put(Ring.oneRing, frodo);
}

@Test
public void basic_assertions_examples() {
    assertThat(frodo.getAge()).isEqualTo(33);
    assertThat(frodo.getName()).isEqualTo("Frodo");
}

// --> New 1.4 FEST ASSERT Feature
@Test
public void isIn_isNotIn_assertions_examples() {
    assertThat(frodo).isIn(fellowshipOfTheRing);
    assertThat(sauron).isNotIn(fellowshipOfTheRing);
}

@Test
public void list_assertions_examples() {
    assertThat(fellowshipOfTheRing).hasSize(9)
        .contains(frodo, sam)
        .excludes(sauron);

    fellowshipOfTheRing.clear();
    assertThat(fellowshipOfTheRing).isEmpty();
}
```

```

    @Test
    public void onProperty_usage() {
        // with simple property
        assertThat(fellowshipOfTheRing).onProperty("name")
            .contains("Boromir", "Gandalf", "Frodo", "Legolas")
            .excludes("Sauron", "Elrond");
        // with simple property on Race
        assertThat(fellowshipOfTheRing).onProperty("race")
            .contains(hobbit, man, elf)
            .excludes(orc);
        // nested property introspection on Race
        assertThat(fellowshipOfTheRing).onProperty("race.name")
            .contains("Hobbit", "Man", "Elf")
            .excludes("Orc");
    }

    @Test
    public void exceptions_assertions_examples() {
        assertThat(fellowshipOfTheRing).hasSize(9);
        try {
            fellowshipOfTheRing.get(9); // arggg!
        } catch (Exception e) {
            assertThat(e).isInstanceOf(IndexOutOfBoundsException.class)
                .hasMessage("Index: 9, Size: 9")
                .hasNoCause();
        }
    }

    @Test
    public void map_assertions_examples() {
        assertThat(ringBearers).hasSize(4)
            .includes(entry(Ring.oneRing, frodo), entry(Ring.nenya,
galadriel))
            .excludes(entry(Ring.oneRing, aragorn));
    }

    @Test
    public void number_assertions_examples() throws Exception {
        assertThat(gandalf.getAge()).isGreaterThan(frodo.getAge());
        File emptyFile = writeFile("emptyFile", "");
        assertThat(emptyFile.length()).isZero();
        // another way to assert file size/length
        assertThat(emptyFile).hasSize(0);
    }

    @Test
    public void file_assertions_examples() throws Exception {
        File xFile = writeFile("xFile", "The Truth Is Out There");
        File xFileClone = writeFile("xFileClone", "The Truth Is Out There");
        assertThat(xFile).exists()
            .isFile()
            .hasSameContentAs(xFileClone)
            .isRelative()
            .hasSize(22);
    }

    // fields initialization
    private final Race hobbit = new Race("Hobbit", false);
    private final Race maia = new Race("Maia", true);
}

```

```
private final Race man = new Race("Man", false);
private final Race elf = new Race("Elf", true);
private final Race dwarf = new Race("Dwarf", false);
private final Race orc = new Race("Orc", false);

private final Character frodo = new Character("Frodo", 33, hobbit);
private final Character sam = new Character("Sam", 38, hobbit);
private final Character merry = new Character("Merry", 36, hobbit);
private final Character pippin = new Character("Pippin", 28, hobbit);
private final Character gandalf = new Character("Gandalf", 2020, maia);
private final Character gimli = new Character("Gimli", 139, dwarf);
private final Character legolas = new Character("Legolas", 1000, elf);
private final Character aragorn = new Character("Aragorn", 87, man);
private final Character boromir = new Character("Boromir", 87, man);
private final Character sauron = new Character("Sauron", 50000, maia);
private final Character galadriel = new Character("Galadriel", 3000, elf);
private final Character elrond = new Character("Elrond", 3000, elf);
```

```
private final List<Character> fellowshipOfTheRing = new ArrayList<Character>();
private final Map<Ring, Character> ringBearers = new HashMap<Ring, Character>();
```